CSCI 310 Advanced Algorithms
Instructor: Dr. Sebastian van Delden
Email: vandeldensa@cofc.edu

ASSIGNMENT 2 – HUFFMAN CODING -INDIVIDUAL WORK ONLY!!!

Implement a text file compression algorithm that utilizes Huffman Coding/Trees. You ***must*** follow the below implementation details.

PROGRAM GUIDELINES THAT MUST BE FOLLOWED

Main Class: Is simply used to call/test Huffman.compress() and Huffman.decompress()

Huffman Class:

- NOTE: You need to figure out how to read/write binary data from/to a file in Java. Sample code that shows how to manipulate bits in Java is provided in class.

- The **public static void compress(String textFileName)** method reads in a text file that can contain any type of alphabet characters, symbols, spaces, newline characters, or numbers. It uses Huffman Coding to generate two files:
    1. A file named textFileName.compressed which is the binary, compressed file using variable length encoded generated from a Huffman Tree.
    2. A text file which includes all the information needed to reconstruct the Huffman Trees during decompression.
        a. This is kind of weird to include this information as a separate (non-compressed) text file, but I think this is a good idea for "learning purposes" making this assignment very doable.
        b. Here is the exact format of this text file where the first line is the number of unique chars encountered/encoded followed by the characters themselves and their frequencies. Note that the sum of all frequencies is the total number of characters in the input text file. **Print newlines as \n and spaces as \s so that we can see them easily.** Example file:

```
5
A 29
c 13
. 78
9 81
\n 44
```

- The **public static void decompress(String textFileName)** method decompresses the binary file. It also reads in the extra text file with the info about the Huffman tree so that it can be recreated.

HuffmanTree Class:

- Implements the HuffmanTree Skeleton Code presented in the notes: the mergeTrees, printAllCodes, getCode, getCharacter, compareTo methods must be implemented, and other methods as needed.

HuffmanNode Class: Implements the HuffmanNode Skeleton Code presented in the notes: the compareTo( ) method.

PriorityQueue<T> Abstract Class and PriorityQueueLinkedList<T> Class:

- Provided for you from assignment 1. No changes needed. You will need to use this in your compress/decompress methods.

Output Report.docx:

- Show the results of your program on 3 different input text files
    - Copy/paste the text files into the report
    - State what the compression ratio was, as well as the output of the text file containing the char frequencies.
    - Your input text files need to vary in size, about: 10 bytes, 100 bytes, and 10,000 bytes.

SUBMISSION DETAILS

Upload a ZIP file to OAKS by the due date. This file must include all .java files and the output report. ***THIS ASSIGNMENT IS INDIVIDUAL WORK***. A score of zero will be assigned to all programs which contained portions of code that have been duplicated.